**UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS**


# TÍTULO: Servicio y componentes para la actualización automática de aplicaciones y recursos.

**MEMORIA INDIVIDUAL PRESENTADA EN OPCIÓN AL TÍTULO DE MASTER EN INFORMÁTICA APLICADA**

**Autor: Ing. Adonis Cesar Legón Campo**
**Tutor: Dr. José Lavandero García**

**Ciudad de la Habana, Julio de 2009**

## DECLARACIÓN JURADA DE AUTORÍA

Yo **Adonis Cesar Legón Campo**, con carné de identidad **83090213301**, declaro que soy el autor principal del resultado que expongo en la presente memoria titulada **Servicio y componentes para la actualización automática de aplicaciones y recursos**, para optar por el título de Máster en Informática Aplicada.

Este trabajo fue desarrollado durante 2 años en colaboración con mis colegas de equipo, los ingenieros: Yoandy Mesa Fiallo, Jorge Landrian García y Yurdik Cervantes Mendosa, quienes participaron en las distintas fases de desarrollo y me reconocen la autoría principal del resultado expuesto en esta memoria.

Finalmente declaro que todo lo anteriormente expuesto se ajusta a la verdad, y asumo la responsabilidad moral y jurídica que se derive de este juramento profesional.

Y para que así conste, firmo la presente declaración jurada de autoría en Ciudad de la Habana a los __ días del mes de _____ del año _____.

**AGRADECIMIENTOS**

A todos mis colegas del equipo de trabajo les estoy muy agradecido. En especial deseo agradecer al Dr. José Lavandero García, quien fungió como tutor de mi formación como máster. Además, deseo agradecer a todos mis compañeros del proyecto Identidad, tanto profesores como estudiantes, quienes también contribuyeron a mi crecimiento profesional y muy en especial a mi querida madre, de quien estaré siempre muy orgulloso, a mi hermano, a mi querida novia y a todos los familiares y amigos, a los que están y a los que ya no, muchas gracias por todo.

## SÍNTESIS

En el marco de la transformación y modernización de la ONIDEX (Oficina Nacional de Identificación y Extranjería), y con el objetivo de automatizar los procesos relacionados con la Identificación, Migración y Extranjería en la República Bolivariana de Venezuela, surge el sistema SAIME, a partir del uso de modernas tecnologías que permiten aumentar la confiabilidad, eficiencia y seguridad en los trámites relacionados con estos procesos y la centralización de la información que en ellos se gestiona. Con este fin el sistema SAIME se compone de un conjunto de aplicaciones distribuidas en las oficinas de la institución en todo el país. Tanto estas aplicaciones como otros componentes, configuraciones y servicios de los distintos nodos que conforman el sistema, pueden estar sujetos a cambios de versión a partir de modificaciones necesarias, detección de errores, la incorporación de nuevas funcionalidades, entre otros.

En este trabajo se presenta el diseño, implementación e implantación de un servicio de actualización de aplicaciones y recursos, además de algunos de los componentes que se definieron para su mejor funcionamiento y utilidad dentro del sistema SAIME, así como para cualquier otro sistema con características similares, permitiendo principalmente lograr el objetivo de mantener actualizados los distintos nodos del sistema y ahorrar recursos materiales para la distribución de estas actualizaciones hacia todas las oficinas del territorio nacional venezolano.

## ÍNDICE DE CONTENIDOS

## INTRODUCCIÓN

Desde hace algunos años, y principalmente con la extensión del uso de las redes de datos e internet, gran parte de los sistemas de software, en alguna medida, han necesitado agregar funcionalidades que les permitan actualizarse ante la creación de nuevas versiones, la emisión de parches de seguridad, o en general ante algún cambio en sus procesos o flujo de información. Estos sistemas o aplicaciones han implementado sus mecanismos de actualización, pero de forma autónoma, es decir, controlando estos procesos de actualización partiendo siempre de las necesidades y las propias funcionalidades para los que fueron creados.

Los mecanismos de actualización, automáticos o no, han estado orientados a resolver los problemas de soporte y mantenimiento de los sistemas y aplicaciones, partiendo de modelos como el de cliente-servidor, para hacer un uso efectivo de las redes de información y evitar costosos traslados de personal calificado para cumplir estos objetivos. La distribución de actualizaciones a través de redes de servidores y clientes, aporta un valor agregado a este tipo de solución, e incorpora los conceptos de administración centralizada y gestión distribuida, ampliamente utilizados en muchos servicios y protocolos de comunicación, aunque no así para procesos de actualización de aplicaciones de propósito general.

El sistema SAIME de la República Bolivariana de Venezuela, compuesto por un conjunto de aplicaciones y servicios, muchos de estos diseñados sobre “aplicaciones de ventana”, las cuales tienen la característica, que las diferencian de las aplicaciones Web, en que al no estar centralizadas, se necesitan mecanismos diferentes para lograr la distribución de nuevas versiones, que permita garantizar la integridad en el funcionamiento de los nodos del sistema y los servicios que estos brindan.

En las primeras etapas de análisis y diseño del sistema SAIME, se identificaron un conjunto de situaciones que podrían presentarse durante el proceso de despliegue e implantación. Estas situaciones estaban relacionadas con la detección y corrección de errores, modificación de los procesos que gestionan las aplicaciones, incorporación de nuevos dispositivos de captura de datos e imágenes, distribución de reportes, entre otros.

Todas estas posibles situaciones identificadas, concluyeron en determinar que en el marco que comprende las características de este sistema, surge la necesidad de desarrollar un mecanismo general de actualización de aplicaciones y recursos, que permita configurar y definir el proceso de actualización para los nodos que lo conformen, además de administrar las actualizaciones que se generen de forma centralizada, y brindar la posibilidad de extender algunas funcionalidades para personalizar los procesos implicados en este flujo.

A partir de la situación que se presenta en las etapas iniciales de desarrollo del sistema y del análisis de las posibles circunstancias que se enfrenten en las etapas posteriores de despliegue e implantación, surge una **Situación Problémica** que se describe a continuación:

- El sistema SAIME, está compuesto por un conjunto de aplicaciones distribuidas en las oficinas de todo el territorio nacional de la República Bolivariana de Venezuela.

- No existe un mecanismo de actualización, que funcione independiente a las aplicaciones en los nodos del sistema, con posibilidades para su: control, configuración, verificación de estado, registro de errores y que brinde soporte para:
  - Actualización de las aplicaciones en etapas de despliegue y funcionamiento estable, donde pueden surgir:
    - Nuevas funcionalidades.
    - Modificaciones en los procesos.
    - Despliegue de nuevos dispositivos externos.
    - Distribución de reportes.
    - Corrección de deficiencias detectadas en las aplicaciones y componentes de los nodos del sistema.
  - Actualización de componentes y servicios en los servidores de oficina.
  - Distribución de archivos para los reportes en las aplicaciones de las oficinas.
  - Distribución de nuevas configuraciones para el funcionamiento de los nodos del sistema.
  - Publicación centralizada de nuevas versiones de aplicaciones y recursos.

Dada la situación descrita anteriormente, se plantea el siguiente **Problema Científico**.

¿Cómo mantener actualizadas las aplicaciones, componentes y recursos de los distintos nodos que conforman el sistema SAIME, que garantice su consistencia e integridad?

Luego del análisis de las posibles situaciones que se puedan presentar relacionadas con la actualización de las aplicaciones que conforman el sistema SAIME en las distintas etapas de su implantación, se definen un Objetivo General y Objetivos Específicos que puedan guiar el desarrollo de la solución propuesta y se describen a continuación:

**Objetivo General**

- Desarrollar una solución para la actualización automática de las aplicaciones y recursos del sistema SAIME.

**Objetivos Específicos**

- Realizar un estudio de los mecanismos de actualización de aplicaciones existentes.
- Definir una arquitectura para la distribución de las nuevas versiones de aplicaciones y recursos del sistema SAIME.
- Desarrollar el servicio automático, los componentes y las herramientas que soporten el proceso de actualización.

**Hipótesis**

Con el desarrollo de una solución compuesta por un servicio de actualización automática y la publicación centralizada de nuevas versiones de aplicaciones y recursos, se podrá mantener actualizados los nodos que conforman el sistema SAIME, para garantizar su consistencia y buen funcionamiento.

Este servicio de actualización automática de aplicaciones y los componentes que lo conforman, le aportan a la institución un considerable ahorro de recursos económicos y humanos, debido a que permite realizar la distribución de las nuevas versiones, funcionalidades o corrección de errores detectados en las distintas fases de desarrollo y despliegue de las aplicaciones y los servicios relacionados con el sistema SAIME, sin tener que realizar tediosos y costos movimientos de técnicos y personal calificado hacia las distintas oficinas que se ubican en todo el territorio nacional de la República Bolivariana de Venezuela. Además permite mantener las aplicaciones en un estado consistente, para gestionar la información que se deriva de los trámites y servicios que ofrece la institución a los ciudadanos del país, así como el control de la generación de actualizaciones de forma centralizada y la verificación del estado y los errores del proceso de actualización en cada nodo del sistema, aumentando su utilidad para los administradores que monitorean y controlan todo el sistema.

Para describir la solución propuesta a la problemática planteada, el contenido de este trabajo se compone de 2 capítulos, los cuales se presentan a continuación:

**Capítulo 1** Estado del Arte: En este capítulo de muestra un investigación sobre el estado en que se encontraban algunos de los sistemas de actualización existentes y se hace referencia a las tecnologías utilizadas para la solución del problema planteado.

**Capítulo 2** Resultado: Se presentan todos los pasos en el desarrollo de la solución propuesta, a través del análisis, diseño, implementación y despliegue, además se muestran algunos de los resultados obtenidos con el sistema ya implantado.

# CAPÍTULO 1: ESTADO DEL ARTE

## 1.1 Sistemas y mecanismos de actualización automática

A continuación se analizan algunos de los principales sistemas, componentes y mecanismos existentes, que realizan procesos de actualización, y la forma en que han servido a la investigación realizada, para proponer la solución posteriormente descrita.

### *1.1.1 Updater Application Block*

Desde el año 2005, Microsoft y su equipo Patterns & Practices [1], han desarrollado un conjunto de componentes de software reusables, con el objetivo de ayudar a aumentar la productividad en la implementación de soluciones de software, creando guías y buenas prácticas para el proceso de desarrollo. Basado en diseños de arquitectura ajustados, para aprovechar las potencialidades de la plataforma de desarrollo .Net, estos componentes está relacionados con aspectos comúnmente utilizados en todo tipo de aplicación, como son el acceso a datos, la interfaz de usuario, criptografía y seguridad, manejo de excepciones, entre otros. Estos componentes de software son conocidos como **Application Blocks** [2] o bloques de aplicación, y algunos de estos se han compuesto para formar la **Enterprise Library** [3], de la que existen implementaciones para las distintas versiones del .Net Framework y con mejoras en sus funcionalidades.

El **Updater Application Block** [4] es uno de estos bloques de aplicación, creados para que una determinada aplicación pueda implementar un proceso de actualización, a través de los pasos de detección, descarga y aplicación de actualizaciones, que han sido publicadas en una ubicación central. Este mecanismo de actualización fue la base del diseño de la solución de actualización automática que se propone en este trabajo, aunque fue necesario hacer un conjunto de modificaciones y re-implementación. También fue necesario agregar algunas funcionalidades que no estaban desarrolladas, principalmente para lograr mantener una red de distribución de actualizaciones, como se explicará en los epígrafes siguientes.

Este bloque de aplicación, junto con todos los demás que existen en las distintas versiones de la Enterprise Library, se han ido mejorando con el tiempo, a partir de la retroalimentación de la comunidad de desarrollo de MSDN y actualmente tiene una mayor cantidad de funcionalidades que no existían en las primeras versiones que fueron analizadas en el año 2005, cuando se comenzó el desarrollo de esta solución de actualización automática.

### *1.1.2 Kaspersky Administration Kit*

El Kit de Administración del Kaspersky Antivirus o **Kaspersky Administration Kit** [5], es una herramienta potente y completa, que permite crear un sistema unificado, distribuido y de control centralizado, para la protección de estaciones de trabajo y servidores contra la amenaza de virus. Es aplicable en redes de distinta complejidad con numerosos nodos, oficinas remotas y usuarios móviles.

Este sistema y su arquitectura, para la distribución de actualizaciones de las bases de datos de detección de virus, se ajustaba a algunos de los requerimientos que se tenían

para el desarrollo de un servicio de actualización automática, capaz de tener varios nodos clientes y servidores, que permitieran un despliegue de topología jerárquica y llevar cierto control en los procesos de actualización, además de utilizar un mecanismo mediante el cual se describen las actualizaciones en documentos XML que se publican.

Esta herramienta, está dotada además, de muchas otras funcionalidades que aumentan el control de los procesos de detección de virus, configuración centralizada, instalación a distancia, monitoreo del estado de protección del los nodos y el manejo de políticas de seguridad en redes corporativas.

## 1.2 Características de los sistemas de actualización automática

En el estudio realizado, se identificó la existencia de dos tendencias principales en la concepción de mecanismos de actualización de aplicaciones (ver Fig. 1). La primera se evidencia con la solución de Microsoft, y está orientada a la publicación de actualizaciones en el servidor, dejando la responsabilidad a los clientes de realizar el proceso de actualización, mientras que en la segunda tendencia, el servidor tiene conocimiento de sus clientes y es capaz de ejecutar operaciones de actualización en estos, aunque también existe la posibilidad de que el cliente decida cuándo actualizarse, de forma manual o automática, esta variante es aplicada por la solución de Kaspersky.

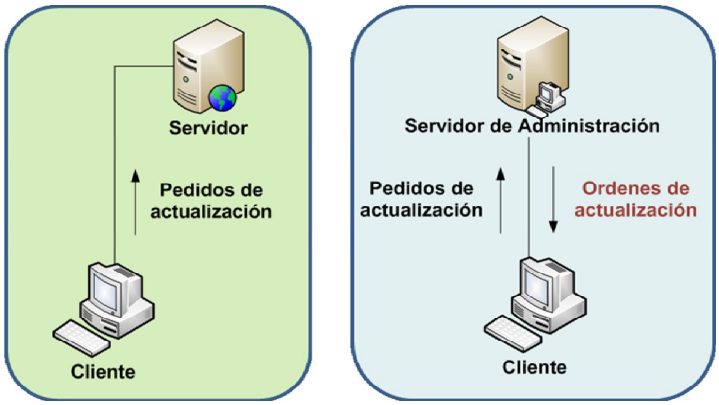


Fig. 1 **–** Tendencias en la concepción de mecanismos de actualización de aplicaciones.

Partiendo del análisis de los anteriores mecanismos de actualización, se pueden identificar algunas características y conceptos comunes que fueron utilizados para el análisis y diseño de la solución propuesta. Entre los principales aspectos se describen:

- Se identifican los pasos: detección, descarga y activación o aplicación, en el proceso de actualización.
- Descripción de los procesos de actualización a través de documentos en formato XML.
- Publicación de los documentos de actualización de forma centralizada.
- Empleo del mecanismo cliente-servidor a través de protocolos de comunicación ampliamente utilizados como HTTP.

Además de estas características comunes se han identificado otras que proveen mayores funcionalidades. Para el caso del *Updater Application Block*, es posible

utilizarlo como componente de actualización en aplicaciones de forma independiente, permitiendo configurar este proceso de forma autónoma y el *Kaspersky Administration Kit* permite configurar una red de distribución y control de actualizaciones a través de distintos servidores y de esta forma propagar la emisión de nuevas versiones de bases para la detección de virus y de nuevas funcionalidades en los nodos cliente.

Todas estas funcionalidades, características y esquemas de arquitectura fueron utilizados como parte de las fases de análisis y diseño que se muestran en los epígrafes posteriores y le aportan conceptos importantes al desarrollo de la solución de actualización automática de aplicaciones, actualmente implantada en el sistema SAIME de la República Bolivariana de Venezuela y con posibilidades de ser utilizado como producto independiente para sistemas semejantes.

## 1.3 Tecnologías utilizadas

La implementación de todos los componentes y del servicio de actualización se realizó sobre la tecnología .Net de Microsoft, con el Microsoft .Net Framework v1.1 y como entorno de desarrollo, el Microsoft Visual Studio .Net 2003. Se debe destacar que al no existir otra tecnología implicada, además del .Net Framework de Microsoft, también es posible usar estos componentes e incluso desarrollar extensiones para el proceso de actualización, utilizando la versión libre de esta plataforma, conocida como Proyecto Mono o cualquier otro proyecto de características similares, sin tener que realizar grandes modificaciones en la estructura o diseño de los componentes y de esta forma serán también portables hacia otras plataformas o Sistemas Operativos como Linux.

Para el desarrollo del servicio de actualización, se utilizó la tecnología “Windows NT Service” [6], que permite la implementación de servicios sobre la plataforma Windows. En otras plataformas que utilizan kernel Linux, existen los llamados “demonios”, para la implementación de servicios, lo cual permitiría el desarrollo de una solución similar.

Además del servicio de actualización, existen un conjunto de componentes dentro de la solución, uno de estos es una aplicación que le permite al usuario comunicarse con el servicio, y para esto se utilizó la tecnología .Net Remoting [7], que permite establecer una comunicación entre distintas aplicaciones o procesos y componentes en una misma estación o de forma distribuida, a través de distintos protocolos de comunicación y mecanismo de serialización de la información transmitida.

La comunicación para la descarga de las actualizaciones, se realiza a través de protocolos muy comúnmente utilizados como el HTTP, de esta forma es posible que existan diferentes plataformas de software entre los nodos clientes y servidores, tal es el caso que para la solución de actualización implementada para el SAIME, se utilizó IIS 6.0 para los nodos servidores en las oficinas, y Apache 2.0 para el servidor de actualizaciones central. Además es posible implementar la descarga de recursos sobre otros protocolos conocidos, como: FTP.

En la modelación de los diagramas del proceso de desarrollo se utilizó el Altova UModel 2008 [8], que es una herramienta de modelado UML, que permite una gran integración con proyectos de Visual Studio .Net para le generación de código, ingeniería inversa y sincronización.

# CAPÍTULO 2: RESULTADO

## 2.1 Requerimientos

Durante la fase de análisis de los requerimientos que se necesitaban para el diseño de este servicio, se tuvieron en cuenta algunas características fundamentales que debían ser cumplidas, de forma tal que la solución se pudiera adaptar a las características distribuidas del sistema y su arquitectura:

***Distribuido:*** este servicio debe ser distribuido por todos los nodos que conforman el sistema, para permitir el despliegue total de nuevas versiones para cada tipo de aplicación, componente, o recursos en general y de esta forma cumplir con su principal objetivo.

***Optimizar el tráfico por la red:*** debe implementar los mecanismos necesarios para no saturar el canal de datos o ancho de banda disponible en los ambientes de conexión, principalmente entre los servidores de las oficinas y el centro de datos, debido a que este mismo canal es compartido con otros servicios y flujos de datos que son imprescindibles para el funcionamiento del sistema, por cada oficina o estación que se conecte directamente con el centro de datos.

***Configurable:*** a través de formatos estándares como XML, debe ser posible configurar todos los nodos que utilizan este servicio, de una forma fácil de comprender e incluso brindando herramientas o aplicaciones gráficas para soportar la gestión de esta información de configuración.

***Actualización múltiple****:* el servicio debe permitir actualizar varias aplicaciones, componentes o carpetas de archivos en un mismo nodo, ya que el sistema permite la existencia de varias aplicaciones en el mismo nodo y todas pueden estar sujetas a modificaciones que necesiten de su actualización.

***Centralizado:*** la emisión de nuevas versiones de las aplicaciones que conforma el sistema se debe hacer solo desde la sede central del mismo para garantizar que el despliegue se haga en cascada y escalonadamente hacia todos los nodos.

***Extensible:*** permitir que se puedan agregar nuevas operaciones a realizar en los clientes como parte del proceso de actualización, una vez que se hayan descargado los recursos de la nueva versión publicada.

***Log de errores:*** como la mayoría de los servicios, se debe persistir en archivos “logs” de texto, los errores que puedan ocurrir en las distintas fases del proceso de actualización.

***Interfaz gráfica:*** mostrar el estado del proceso de actualización mediante una interfaz gráfica hacia el usuario u operador del sistema y algún mecanismo o componente para poder controlar el servicio y conocer de su estado de ejecución actual de forma amigable.

***Auto-actualizable:*** el servicio debe proveer un mecanismo o componente para actualizarse a sí mismo, de esta forma se podrán generar nuevas versiones del propio servicio o de alguno de sus componentes, en el caso que sea necesario.

## 2.2 Diseño

Antes de comenzar el desarrollo del servicio de actualización, se modeló e implementó el principal componente que conforma la solución final, el cual debía tener toda la lógica que le permitiera a una aplicación independiente, implementar su propio mecanismo de actualización automática, utilizando la metodología y configuración definidas para su proceso de actualización.

Para el desarrollo del componente de actualización se siguió el mecanismo de comunicación cliente-servidor, mediante el cual la información es publicada y administrada en un servidor y los nodos cliente interactúan con este haciendo pedidos de recursos, en este caso, el mecanismo se utiliza para la comprobación de actualizaciones y para el propio proceso ya en ejecución.

Las nuevas versiones de actualizaciones son comprobadas a través de archivos en formato XML que describen qué se debe hacer en el proceso de actualización, además de contener la información de los recursos publicados, para que el cliente pueda determinar las nuevas versiones e incluso cuál recurso es diferente en el nodo cliente respecto al que se encuentra publicado en el servidor, y así poder decidir si lo debe descargar o no. Este proceso de actualización en el nodo cliente, está definido en tres etapas fundamentales: **Verificación**, **Descarga** y **Activación**, que serán explicadas en próximos epígrafes.

El diagrama que se muestra en el Anexo 1, representa las principales clases que conforman el diseño del componente fundamental para todo el proceso de actualización.

Entre estas clases se deben destacar: el **AdministradorActualizaciones**, que se encarga de controlar todo el proceso de actualización, a través de varios “controladores”, que realizan las operaciones específicas por cada proceso. Estos controladores, con clases que ejecutan y manejan cada parte del proceso de actualización, estos son: el **ControladorVerificacion** que gestiona el proceso de verificar la existencia de nuevas actualizaciones, el **ControladorDescarga** que permite realizar el proceso de descarga utilizando el descargador que se configure en el nodo cliente, el cual es un clase que implementa la interfaz **IDescargador**, y el **ControladorActivacion**, que gestiona la activación de las actualizaciones, ejecutado el método “Activar” de cada activador que se defina por configuración, los cuales implementan la interfaz **IActivador**, y utilizan sus propias configuraciones específicas, también definidas en los archivos de configuración del nodo cliente.

Las clases **Documento** y **Tarea**, pueden ser serializables, lo cual permite que sea más fácil el proceso de persistencia y carga de los documentos de actualización y las tareas pendientes dentro del proceso, a través del mecanismo de serialización XML que provee la tecnología .Net o disponible en otras tecnologías y plataformas de desarrollo y que es ampliamente utilizando en las aplicaciones en la actualidad.

## 2.3 Proceso de actualización

A continuación se explica el proceso de actualización que se desencadena en el cliente, desde que se verifica la posible nueva actualización, hasta que se termina de **aplicar** y **activar**.

### *2.3.1 Verificación*

Este proceso es el primero que ocurre cuando comienza la actualización en un nodo del sistema. Una vez que el servicio se vuelve a activar, luego de haber estado un intervalo de tiempo "suspendido", comienza el proceso verificando que no haya actualizaciones pendientes o en espera, lo cual puede ocurrir por cualquier error en algún paso del proceso de actualización anterior al actual. Aquí aparece una característica de este servicio y es la posibilidad de "reiniciar" actualizaciones que hayan quedado pendientes por error. El error más frecuente ocurre cuando existen problemas en la conexión de red, de esta forma cuando se recupere la conexión se puede continuar con el proceso de actualización pendiente.

En el cliente se guarda la información de la tarea pendiente en archivos de extensión "**.tarea**", que contienen lo necesario para que el componente de actualización sepa en qué paso del proceso se detuvo y qué es lo que faltó por procesar. La cantidad de veces que se desee "reaplicar" actualizaciones pendientes, se puede especificar en la configuración del nodo cliente. Cada vez que se intente aplicar la actualización pendiente, y no haya terminado por alguna razón, se descuenta la cantidad de veces para esa tarea y cuando llega a cero se comienza el proceso desde el principio. Si no se quiere usar esta funcionalidad se pone el valor de la configuración a -1.

Una vez que no queden tareas pendientes, se pasa a las descarga de los **documentos**, y a la verificación de nuevas actualizaciones disponibles para la aplicación específica. Esto se hace comparando la información de la nueva actualización descrita en los documentos descargados con las que ya se han aplicado en el cliente. Durante este proceso de determinación de nuevas actualizaciones, también se hace un análisis de cuáles recursos se han modificado en la estación donde corre el componente de actualización, comparando con los recursos que se encuentran en el servidor, para solo descargar los nuevos o los que han cambiado, y de esa forma contribuir a evitar un problema fundamental que es el tráfico en las redes, con el objetivo de no congestionar estas con descargas innecesarias.

### *2.3.2 Descarga*

Para realizar la descarga de recursos, se ha definido a nivel de diseño de clases, una interfaz general de implementación, que describe las funcionalidades que debe proveer un "**descargador**" en específico, y así poder extender el proceso de descarga a otros componentes personalizados, según las necesidades del sistema, incluso para el caso en que se desee utilizar otros protocolos de comunicación. Esta interfaz se llama **IDescargador**. Actualmente en la biblioteca de clases del componente de actualización se encuentra disponible un descargador sobre protocolo HTTP de forma muy simple, aunque se pudiera desarrollar sobre otros mecanismos más avanzados de descarga, que implementen incluso más seguridad.

En este paso del flujo, simplemente se obtienen los recursos requeridos en el proceso de actualización, descargándolos desde el nodo servidor y se ubican en una carpeta por defecto que el componente ya ha definido, organizado en carpetas con el identificador de cada aplicación como nombre.

### *2.3.3 Activación*

Este es el último proceso que tiene lugar en el flujo de actualización. En este paso se define exactamente qué operaciones se realizarán con los recursos ya descargados de la nueva versión de la aplicación. La información sobre qué hacer en el paso llamado **"activación"**, se describe en los documentos de actualización, que fueron generados en el servidor central, y descargados en los nodos cliente, durante el proceso de verificación.

Para la activación también se ha definido una interfaz que describe la funcionalidad de un "**activador**" de los existentes en el componente de actualización, u otro que se quiera implementar, teniendo en cuenta que el principal objetivo de un activador, es realizar algún procesamiento en el nodo donde se está ejecutando la actualización de determinada aplicación, sobre los recursos que fueron descargados en el proceso anterior. La interfaz del activador se llama "**IActivador**".

El componente de actualización ya tiene algunos de los activadores que realizan las tareas más frecuentemente usadas, como:

- copiar los recursos de una ubicación para otra.
- verificar los archivos descargados usando funciones resumen (hash).
- borrar archivos en una determinada ubicación.

Para poder implementar más activadores de actualizaciones, solo se debe especificar en los documentos cuáles son, y poner en los nodos clientes de actualización, los ensamblados que contienen las clases que implementan la interfaz "**IActivador**".

Es muy frecuente que en los procesos de descarga y activación ocurran errores para los cuáles el sistema aborta la operación, y pone esta actualización como pendiente para ser resumida después, de esta forma el administrador o personal de soporte, puede inspeccionar en un cliente en los archivos "**.tareas**" y deducir cuáles son los errores que pudieran haber ocurrido durante el proceso, para poder actuar y resolver el problema.

El diagrama que se muestra en el Anexo 2, describe los pasos del mecanismo de actualización.

## 2.4 Arquitectura y configuración

A continuación se describen la arquitectura y las configuraciones necesarias para que todo el proceso de actualización funcione correctamente a través de los distintos nodos del sistema, tanto servidores y clientes como un tipo de nodo llamado "**intermediario**", este último permite propagar las actualizaciones generadas centralmente hacia otro nivel de la red de distribución de actualizaciones del sistema.

### *2.4.1 Nodo servidor*

Es necesario en primer lugar configurar el nodo servidor para emitir y publicar las actualizaciones. Para generar las actualizaciones de cada aplicación, recursos o servicio del sistema, se utiliza un componente que forma parte del paquete de componentes de la actualización y se presenta como utilidad o herramienta, el cual es posible emplear en una aplicación desarrollada por el sistema específico que utilice esta solución de actualización automática. En el caso del sistema SAIME, se desarrolló una funcionalidad dentro del módulo de Administración Global que les permite a los administradores gestionar y controlar la generación y publicación de actualizaciones para cada aplicación del sistema.

Las actualizaciones, tanto los propios archivos que se van a distribuir de la nueva versión, como los documentos (archivos en formato XML con extensión "**.docum**") que describen el proceso de actualización en los nodos cliente, son publicados en un servidor Web sin requerimientos de tecnología o plataforma específicas, debido a que solo es necesario que los nodos cliente pueda descargar estos recursos. El siguiente ejemplo muestra en un caso real para el sistema SAIME, la forma en que se publican estos recursos.

Para el caso de la aplicación de Oficina de Captación, se publica a partir de la dirección del servidor central http://updateserver:8080/**oficina** (como se puede notar la URL termina en el nombre de la aplicación) y seguido por la carpeta "**app**" que contiene la aplicación y en la misma ubicación raíz, la carpeta "**docs**" que contiene los "**documentos**" de la actualización, de esta forma queda definido:

- Para la aplicación: http://updateserver:8080/oficina/app
- Para los documentos: http://updateserver:8080/oficina/docs

Cada documento de actualización posee un identificador único de la aplicación que será objetivo del proceso de actualización específico. Los documentos también pueden contener otro documentos referenciados o incluidos para ampliar las operaciones en una misma actualización, estos otros documentos representan a los archivos que se encuentran en otras carpetas dentro de la carpeta de la aplicación que se va a actualizar, es decir cuando se genera un nueva actualización de una aplicación, se crea un documento (archivo ".docum") que es el principal y tantos archivos de documentos incluidos en este, como carpetas tenga la aplicación. Por esta razón se definió, que en las configuraciones para cualquier proceso de actualización existente en el sistema, siempre se genere un documento denominado "**principal**", el cual tiene el mismo nombre de la aplicación que se va a actualizar, y este a su vez incluye a los demás documentos de las otras carpetas en los recursos de la aplicación.

Por tanto el documento principal para la actualización de la aplicación de Oficina de Captación sería:

- http://updateserver:8080/oficina/docs/oficina.docum

### *2.4.2 Nodo cliente*

En un nodo cliente que tenga instalado el servicio de actualización, también se deben hacer determinadas configuraciones, a través de un archivo XML que describe cuál o cuáles son los documentos que debe utilizar para la actualización de cada aplicación o carpeta de recursos, es decir, si en una estación del sistema existe más de una aplicación, en la configuración del servicio se especifica la ubicación de cada archivo de configuración para cada aplicación en el nodo cliente.

Estos archivos de configuración para cada aplicación en el nodo cliente, contienen toda la información necesaria en aspectos como: la ubicación en el disco de la carpeta donde se deben copiar los recursos y, si es una aplicación, el nombre del ejecutable de forma tal que el servicio pueda verificar si está corriendo y mostrar un mensaje de advertencia al operador que se encuentre trabajando en ese momento. Otras informaciones como las credenciales para la conexión con el servidor Web y el tipo de descargador que se va a usar, además de el intervalo de tiempo que se debe esperar entre una verificación y otra especificado en segundos, y la cantidad de veces que se desea intentar reaplicar las tareas de actualización pendientes son algunas de las configuración que se establecen para cada aplicación que se quiera actualizar en el nodo cliente.

### *2.4.3 Nodo cliente intermediario*

En un determinado nodo cliente de la red de distribución de actualizaciones, es posible definir, además de las configuraciones ya explicadas, si este nodo será un cliente intermediario. El intermediario es otro tipo de nodo cliente, pero tiene la característica de permitir "**propagar**" la actualización que realizó. De esta forma este tipo de nodo, realiza todo el proceso de generar los nuevos documentos y publicarlos junto con los nuevos recursos de la actualización que descargó, y así queda listo para que otro nodo cliente, normal o intermediario, pueda actualizarse y formar parte de la red.

El diagrama que se muestra en el Anexo 3, describe la arquitectura del sistema o red de actualización en el caso de cómo se ha aplicado para las oficinas regionales del sistema SAIME.

## 2.5 Componentes

A continuación se describen los principales componentes que conforman esta solución de actualización automática.

### *2.5.1 Servicio*

El servicio de actualización automática es en sí un "Windows NT Service" o servicio de Windows, que corre constantemente desde que inicia la estación de trabajo, y verifica cada cierto tiempo, (definido por configuración para cada **cliente de actualización** en la misma estación) la existencia de nuevas actualizaciones. Este servicio corre con los privilegios de un usuario que el Sistema Operativo define por defecto llamado "LOCAL SERVICE", que tiene permisos pare leer y escribir en determinadas ubicaciones dentro del sistema.

La configuración del servicio describe los distintos clientes de actualización que están configurados en un nodo del sistema. Cada vez que se detectan actualizaciones nuevas el servicio las aplica por defecto aunque se le pudiera preguntar al usuario si quiere ejecutarla o no, y si es una aplicación la que se va a actualizar entonces se le muestra una ventana sugiriéndole al usuario que debe cerrar la aplicación para efectuar el proceso.

### *2.5.2 Controlador servicio*

Como parte del paquete de componentes de la actualización automática, existe una aplicación que se ejecuta por separado del servicio de actualización, llamada "Controlador de Servicio", la cual se encarga de permitirle al operador en la estación de trabajo, poder controlar el servicio de actualización de forma amigable, a través de un icono que aparece en la barra de tareas del escritorio y que muestra la imagen logo del sistema SAIME. Esta funcionalidad es común en muchas aplicaciones que funcionan como servicios y que necesitan de una interfaz gráfica, para que el usuario le pueda ejecutar operaciones, como es el caso de los antivirus.

Esta aplicación permite realizar un conjunto de operaciones de control y verificación de estado (a través de un menú contextual), sobre el servicio que se está ejecutando, las cuales se describen a continuación:

- **Mostrar errores**: ejecuta el editor de texto definido por defecto en el Sistema Operativo del nodo cliente y muestra como contenido el archivo "log" con todos los errores ocurridos en el historial de funcionamiento del servicio.

Luego aparece un listado con una opción de menú por cada cliente de actualización que se haya configurado en la estación. De cada uno se despliega otro menú que muestra las opciones:

- **Actualizar**: para ejecutar el proceso en el cliente de actualización seleccionado, lo cual es útil para el caso en que se sabe que ya hay una actualización publicada pero no se ha actualizado el cliente porque tiene que esperar a que se venza el tiempo de espera entre una actualización y la siguiente.
- **Actualizaciones**: esta opción muestra en el editor de texto configurado por defecto en el Sistema Operativo, el listado de las fechas de las actualizaciones efectuadas para un cliente de actualización.
- **Configurar**: con esta opción se ejecutará una aplicación que viene incluida con la instalación del servicio de actualización y que forma parte del paquete de componentes, permitiendo configurar un cliente de actualización.

### *2.5.3 Actualizador servicio*

Uno de los principales requerimientos para esta solución de actualización automática, era brindar el soporte necesario, para que el propio servicio y sus componentes, se pudieran actualizar de la misma forma que las demás aplicaciones y recursos del nodo cliente donde estuviera ejecutándose.

Para dar solución a este problema, se diseñó e implementó otro componente, que consiste en una aplicación sencilla, sin interfaz gráfica, que una vez configurada en el

nodo cliente para que pueda ser descargada y ejecutada como otra aplicación, cuando se genera una nueva actualización del propio servicio, permitiera cumplir este objetivo. Además esta aplicación llamada “Actualizador Servicio”, permite realizar algunos ajustes en la corrección de errores, o para modificar aspectos de las configuraciones, en los nodos clientes del sistema, aunque solo para casos en los que sea necesario.

### *2.5.4 Utilidades*

Los siguientes componentes representan algunas de las utilidades que brinda este paquete de actualización, los cuales han sido aplicados en varias situaciones dentro de la solución de actualización automática para el sistema SAIME.

#### 2.5.4.1 Aplicaciones para la gestión de las configuraciones

Entre los componentes que conforman esta solución, se incluyen dos aplicaciones de escritorio, una que permite gestionar los archivos de configuración para un nodo cliente <EditorConfiguracionCliente.exe>, y otra que gestiona la generación de actualizaciones para luego ser publicadas centralmente a todos los nodos del sistema <EditorDocumentos.exe>. Las aplicaciones son sencillas y usan algunas del las herramientas que provee este producto, y para el caso de sistema SAIME, están vinculadas a las opciones que se muestran al usuario, en el icono del controlador de actualizaciones.

#### 2.5.4.2 Agente actualizador

Este agente actualizador consiste en una clase que implementa el mecanismo para ejecutar el proceso de actualización, a partir de la configuración que se establezca sobre las aplicaciones o recursos que se quieran actualizar en un nodo cliente. De esta forma el agente pueda ser usado desde una aplicación en específico para controlar, y ejecutar el proceso de actualización, en caso que sea necesario usarlo de forma personalizada y no a través del servicio de actualización.

#### 2.5.4.3 Generador de actualizaciones

El generador de actualizaciones, es una clase que forma parte de las utilidades incluidas en los componentes de esta solución, que permite generar los documentos de actualización para una aplicación o recursos en general. Esta utilidad permite el desarrollo de una aplicación para la generación centralizada en las actualizaciones del sistema, con la cual se podría llevar un control de estas aplicaciones y el registro de las versiones que ya han sido publicadas.

En el caso del sistema SAIME, existe una funcionalidad en el módulo de Administración Global, que emplea esta utilidad para generar las actualizaciones de las aplicaciones y registra esta información para un mejor control en este proceso.

#### 2.5.4.4 Publicador de actualizaciones

Luego de generar los documentos de actualización, para una aplicación específica, es necesario publicarlos o subirlos (upload), a través de un determinado protocolo que se

defina, por ejemplo FTP. Una de las utilidades que provee estos componentes, llamada publicador de actualizaciones, permite gestionar la publicación de recursos a través de distintos protocolos, aunque actualmente solo soporta FTP sobre SSH, o SFTP, la clase **Publicador** define la interfaz que se debe implementar para extender esta funcionalidad, con el uso de otros protocolos.

En el módulo de Administración Global del sistema SAIME utiliza este publicador para subir los recursos de la nueva versión de la aplicación y los documentos que describen la actualización, hacia un servidor SFTP, facilitando el trabajo a los administradores del sistema.

En el diagrama que se muestra en el Anexo 4, se describen los componentes que conforman esta solución de actualización automática, mediante las descripciones del diseño de componentes y la arquitectura definidas para una solución que pueda ser aplicable de forma independiente a las aplicaciones de un sistema.

En este esquema se muestran los componentes anteriormente descritos, distribuidos en los nodos cliente y servidor que conforman el despliegue básico de esta solución de actualización. El componente <Actualizacion.dll> es el núcleo principal para los procesos de actualización, del cual según se muestra en el diagrama dependen todos los demás componentes y se instala en el nodo cliente. La aplicación <ActualizadorServicio.exe>, no depende de ningún componente, debido a que solo es una aplicación que permite actualizar el propio servicio de actualización.

## 2.6 Resultados

El servicio de actualización automática, se encuentra actualmente desplegado y funcionando en: los servidores de las 42 oficinas de captación de pasaporte, regionales y municipales (con nodos intermediarios), y en las alrededor de 11 estaciones que utilizan los operadores del sistema por cada una de estas oficinas de captación, además se encuentra implantado en 4 puntos fronterizos, 8 aeropuertos, y en el aeropuerto de Maiquetía que funciona con un servidor de oficina y nodos cliente, y que conforman las oficinas para el control migratorio, en 2 oficinas de sede central y en la oficina de personalización de documentos. Todo esto conforma un total de 43 nodos intermediarios, 473 nodos cliente conectados a estos intermediarios y 15 nodos cliente conectados al servidor de actualizaciones principal. Toda la información es transmitida a través de una VPN que conecta a todas las oficinas del país, para garantizar la seguridad y confidencialidad de los datos.

Como se muestra en la tabla del Anexo 5, este servicio está ampliamente probado en el *campo* y lleva aproximadamente 4 años en explotación, soportando la generación de actualizaciones para las distintas aplicaciones de los tipos de oficina que conforman el sistema. Además este servicio ha sido probado y utilizado en otros proyectos de la Universidad con el Ministerio del Poder Popular para las Relaciones de Interior y Justicia de la República Bolivariana de Venezuela, como es el caso de Registros y Notarias, que presenta un esquema de distribución y despliegue semejante al SAIME.

## CONCLUSIONES

Con el desarrollo de este trabajo y teniendo en cuenta los resultados obtenidos, se ha podido demostrar cómo se han cumplido los objetivos principales definidos para la creación de un servicio de actualización automática de aplicaciones y recursos, además de algunos de sus principales componentes, que le dan un valor agregado y mayores posibilidades de aplicación en diferentes entornos.

Es importante destacar que además de proponer una solución de actualización automática, que pueda ser presentada como un producto independiente para ser usado en distintos ambientes con características semejantes, su aplicación directa en el sistema SAIME, a partir de sus dimensiones en cuanto a cantidad de nodos y aplicaciones que lo conforman, ha permitido tener un entorno de pruebas, que ayuda a la retroalimentación de posibles errores y mejoras en la solución final, y de esta forma ir perfeccionando sus funcionalidades y prestaciones.

Entre las principales características que se pueden describir de este producto desarrollado se encuentran:

- Permite la generación centralizada de las actualizaciones de aplicaciones y recursos, a través de documentos en formato XML que describen el proceso para ser ejecutado por los clientes.
- Es posible configurar a los clientes como “clientes intermediarios”, para crear una red de distribución de actualizaciones a varios niveles según se defina.
- En el proceso de detección de nuevas actualizaciones, se utiliza un mecanismo para verificar el resumen o “hash” de los recursos en el cliente, para no realizar descargas innecesarias y disminuir el tráfico en la red.
- Se permite actualizar varias aplicaciones en el cliente según se configure.
- A través de la implementación de las interfaces definidas, es posible agregar nuevos tipo de cliente y protocolos de descarga, así como procesos de activación de las actualizaciones.
- Se incluyen algunos componentes que permiten la generación y publicación de actualizaciones, la configuración de los clientes y el control del servicio a través de una interfaz de usuario.
- Existe un mecanismo mediante el cual el servicio es capaz de actualizarse a sí mismo, en caso de que surjan nuevas funcionalidades.

## RECOMENDACIONES

Para siguientes fases de desarrollo de esta solución de actualización automática de aplicaciones, se han definido un conjunto de funcionalidades y herramientas que se recomienda implementar, y de esta forma aumentar su utilidad en los distintos entornos y condiciones donde se pueda desplegar, así como lograr mayores facilidades al usuario administrador, para el control y configuración de los nodos de actualización.

- Terminar el desarrollo para el proceso de publicación y distribución de actualizaciones, en formato compactado, permitiendo optimizar, aún más, el tráfico de recursos de actualización en la red.
- Implementar una herramienta para el control global de todos los clientes e intermediarios en la red de distribución de actualizaciones, y tener información del estado de actualización por cada tipo de aplicación o recurso que se configure en el cliente, utilizando un mecanismo propio de comunicación o a través de protocolos estándares, como el SNMP [9], que permite integrarse con otros sistemas de administración de redes y servicios.
- Desarrollar otros clientes de descarga para brindar posibilidades de publicación de actualizaciones por otros protocolos de comunicación.
- Probar hacer una propuesta de esta solución de actualización sobre una plataforma Linux, utilizando la implementación libre de la tecnología .Net llamada proyecto Mono.

## BIBLIOGRAFÍA

[1] Microsoft. Patterns & Practices. Disponible en
http://msdn.microsoft.com/en-us/practices/bb969103.aspx

[2] Wikipedia. Application Blocks, 2008. Disponible en
http://en.wikipedia.org/wiki/Microsoft_Enterprise_Library#Application_Blocks

[3] Microsoft. Microsoft Enterprise Library, 2008. Disponible en
http://msdn.microsoft.com/en-us/library/cc467894.aspx

[4] Microsoft. Updater Application Block, 2005. Disponible en:
http://msdn.microsoft.com/en-us/library/ms978545.aspx

[5] Kaspersky, Kaspersky Administration Kit, 2003. Disponible en:
http://www.kaspersky.com/sp/administration_kit

[6] Microsoft. Introduction to Windows Service Applications, 2008. Disponible en:
http://msdn2.microsoft.com/en-us/library/d56de412(VS.80).aspx

[7] Microsoft. .NET Remoting Overview. Disponible en:
http://msdn.microsoft.com/en-us/library/kwdt6w2k(VS.71).aspx

[8] Altova, UModel - UML tool for software modeling and application development , 2008. Disponible en: http://www.altova.com/products/umodel/uml_tool.html

[9] Cisco Systems, Simple Network Management Protocol, 1999. Disponible en:
http://www.cisco.com/en/US/docs/internetworking/technology/handbook/SNMP.html

# ANEXOS

## Anexo 1

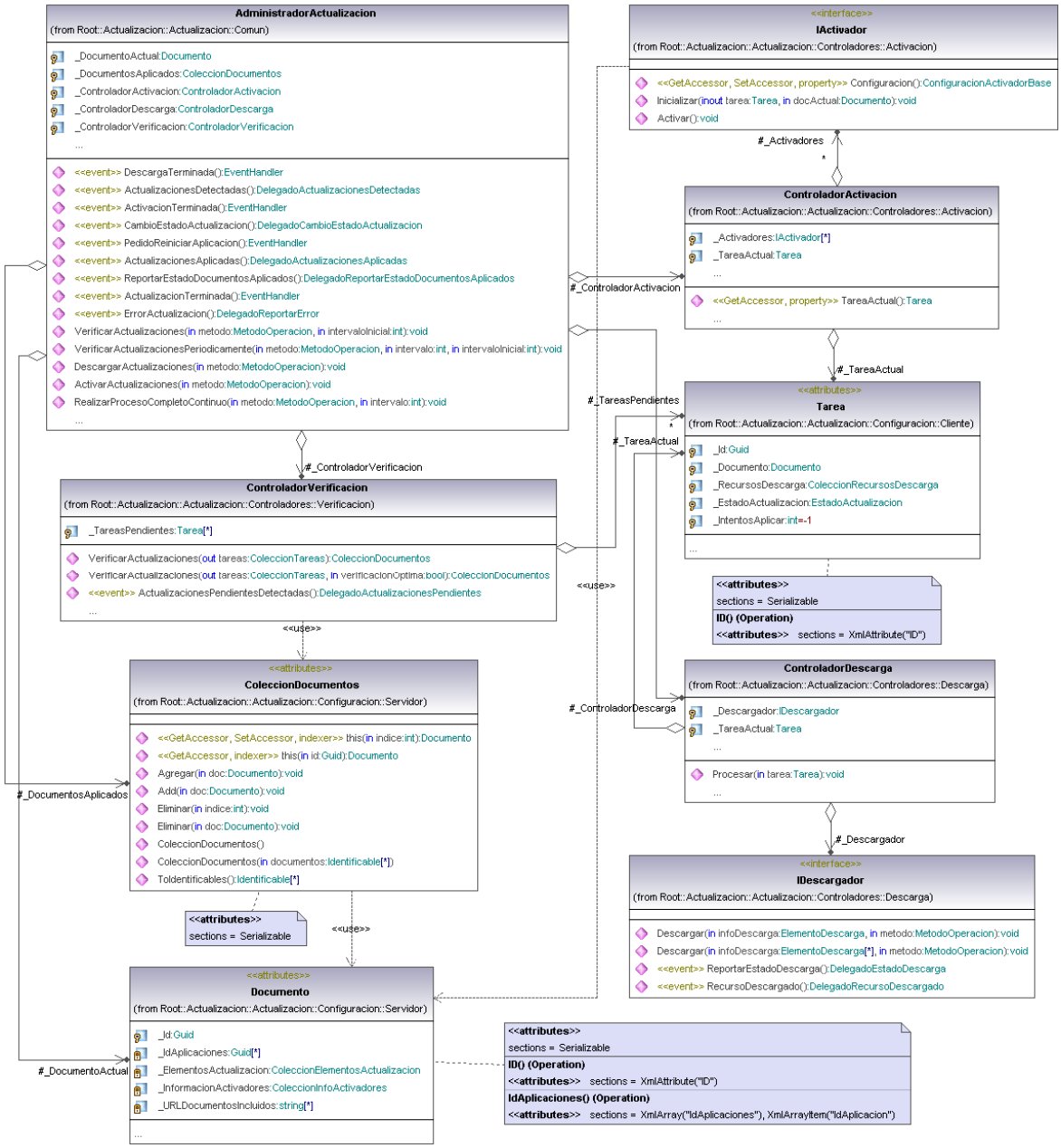


Fig. 2 **–** Diagrama de clases del componente *núcleo* de actualización.

## Anexo 2

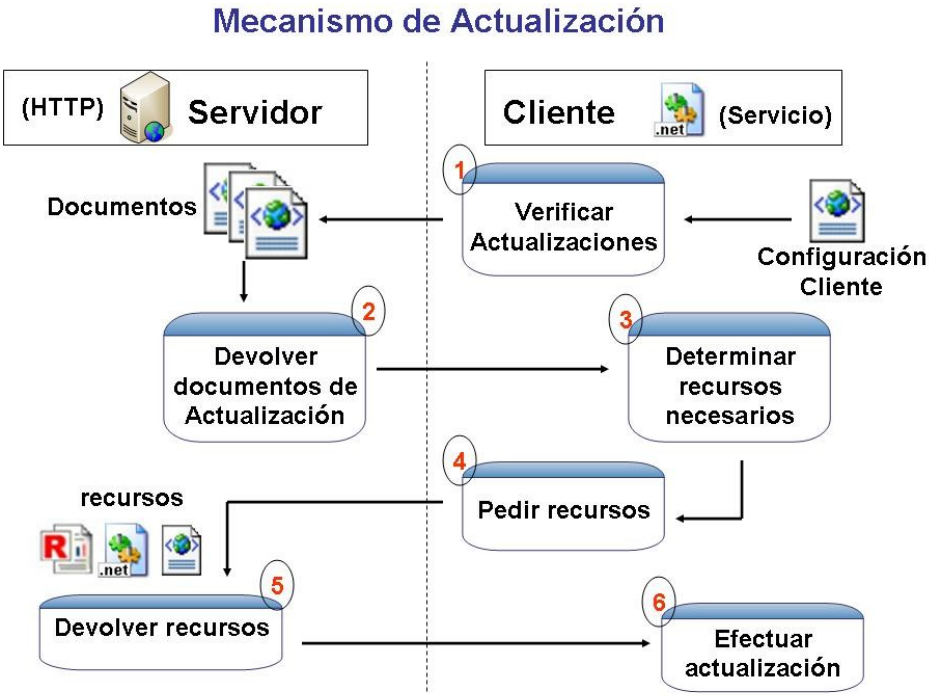


Fig. 3 **–** Diagrama general del proceso de actualización.

## Anexo 3

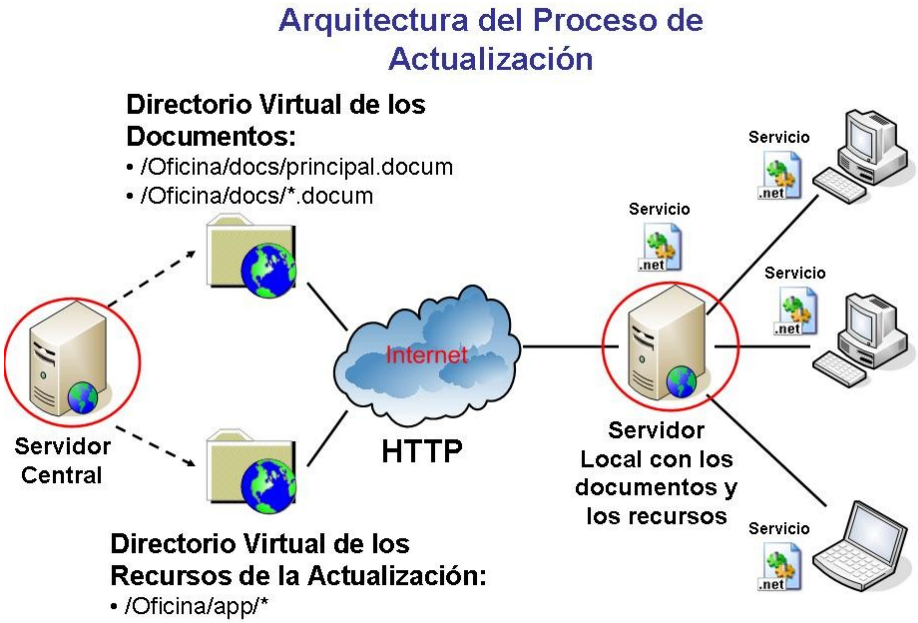


Fig. 4 **–** Diagrama de la arquitectura del proceso de actualización en un ambiente distribuido (Ejemplo: SAIME).

## Anexo 4

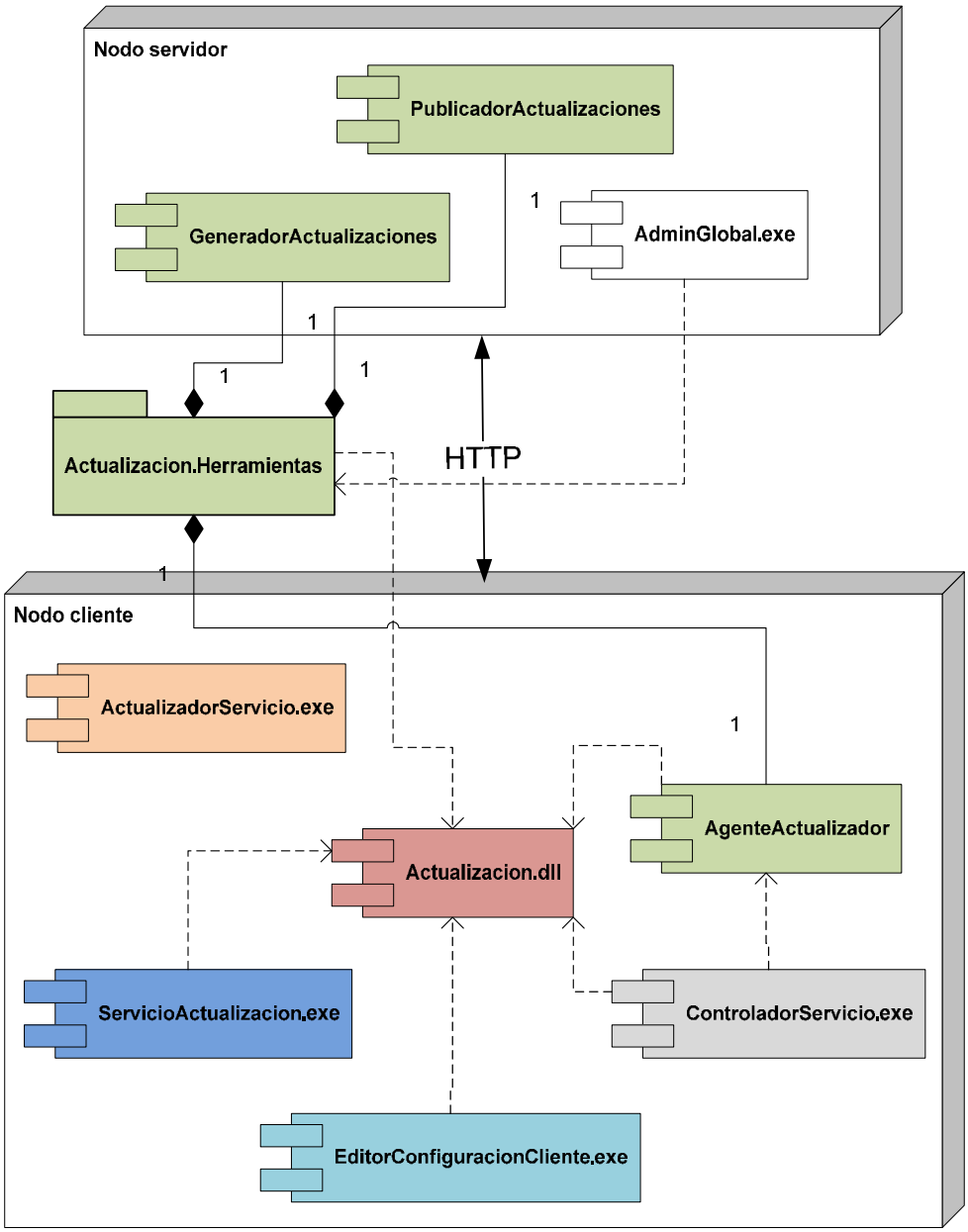


Fig. 5 – Diagrama de componentes y despliegue para la solución de actualización automática.

## Anexo 5

| APLICACIÓN | ACTUALIZACIONES | PRIMERA | ÚLTIMA |
|---|---|---|---|
| Sistema de pasaporte temporal | 16 | 21/01/2007 | 5/11/2007 |
| CPID | 14 | 29/03/2007 | 15/04/2008 |
| Control migratorio en puertos | 1 | 19/09/2007 | 19/09/2007 |
| Aplicación de oficina regional | 61 | 2/3/2007 | **28/07/2008** |
| Control migratorio en puntos fronterizos | 2 | 29/05/2007 | 22/06/2007 |
| Configuración servicio AIMID | 36 | 11/3/2007 | 18/03/2008 |
| Control migratorio en aeropuertos | 5 | 23/03/2007 | 25/04/2008 |
| Conversión decadactilar | 10 | **6/11/2006** | 21/01/2008 |
| Administración Global | 10 | 3/3/2007 | 18/07/2008 |
| Servicio de actualización automática de aplicaciones | 2 | 29/03/2007 | 15/08/2007 |
| Móvil de cedulación | 5 | 11/7/2008 | 17/07/2008 |
| Sede central | 43 | 13/02/2007 | 15/05/2008 |
| Configuración Booking | 2 | 4/6/2007 | 31/07/2007 |

Tabla 1.- Reporte de las actualizaciones emitidas en el sistema SAIME hasta la actualidad.